P⊙RTAL

USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ⦿ The ACM Digital Library ○ The Guide

Searching within **The ACM Digital Library** with **Advanced Search**: (Abstract:"intermediate language" and Abstract:compiler) (start a new search)
Found **47** of 254,384

REFINE YOUR SEARCH

Search Results   Related Journals   Related Magazines   Related SIGs
Related Conferences

▾ Refine by Keywords

Discovered Terms

▾ Refine by People
Names
Institutions
Authors
Reviewers

▾ Refine by
Publications
Publication Year
Publication Names
ACM Publications
All Publications
Content Formats
Publishers

▾ Refine by
Conferences
Sponsors
Events
Proceeding Series

ADVANCED SEARCH
🔍 Advanced Search

FEEDBACK
✒ Please provide us with feedback

Found **47** of **254,384**

Results 1 - 20 of 47

Sort by [relevance ▾] in [expanded form]

💾 Save results to a Binder

Result page: **1**   2   3   next

1 Optimizing Ada on the fly
Sheri J. Bernstein, Robert S. Duff
September 1999 **SIGAda '99:** Proceedings of the 1999 annual ACM SIGAda international conference on Ada

Publisher: ACM ◇ Request Permissions

Full text available: 📄 Pdf (774.62 KB)   Additional Information: full citation, abstract, references, index terms

Bibliometrics: Downloads (6 Weeks): 2,   Downloads (12 Months): 13,   Citation Count: 0

One of the features that makes Ada such a reliable programming language is its use run-time constraint checks. However, such checks slow program execution and increa program size. The AdaMagic™ compiler uses one-pass, on-the-fly flow analysis ...

Keywords: Ada, array bounds checking, check elimination, compiler, optimization, range checking, range propagation, uninitialized variables, warnings

Also published in:
September 1999 **SIGAda Ada Letters** Volume XIX Issue 3

2 The EPN and ESN notations
Irving B. Elliott
July 1984 **SIGPLAN Notices**, Volume 19 Issue 7

Publisher: ACM

Full text available: 📄 Pdf (850.90 KB)   Additional Information: full citation, abstract, references

Bibliometrics: Downloads (6 Weeks): 4,   Downloads (12 Months): 8,   Citation Count: 5

Two specification notations are introduced: (1) An Executable Program Notation (EPN) that can be used either for specifying procedural coding or can serve as an intermedi language produced from a higher-order language. The notation is configured such ...

3 Vortex: an optimizing compiler for object-oriented languages
Jeffrey Dean, Greg DeFouw, David Grove, Vassily Litvinov, Craig Chambers
October 1996 **OOPSLA '96:** Proceedings of the 11th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications

Publisher: ACM ◇ Request Permissions

Full text available: 📄 Pdf (2.45 MB)   Additional Information: full citation, abstract, references, cited by, in terms

Bibliometrics: Downloads (6 Weeks): 4,   Downloads (12 Months): 27,   Citation Count: 49

Previously, techniques such as class hierarchy analysis and profile-guided receiver cla prediction have been demonstrated to greatly improve the performance of application written in pure object-oriented languages, but the degree to which these results ...

Also published in:
October 1996 **SIGPLAN Notices** Volume 31 Issue 10

4 Compiling standard ML to Java bytecodes
Nick Benton, Andrew Kennedy, George Russell
January 1999 **I CFP '98:** Proceedings of the third ACM SIGPLAN international conference
Functional programming
Publisher: ACM ◇ Request Permissions

Full text available: 📄 Pdf (1.54 MB)   Additional Information: full citation, abstract, references, cited by, in
terms

Bibliometrics: Downloads (6 Weeks): 2,   Downloads (12 Months): 49,   Citation Count: 31

MLJ compiles SML'97 into verifier-compliant Java byte-codes. Its features include typ
checked interlanguage working extensions which allow ML and Java code to call each
other, automatic recompilation management, compact compiled code and runtime
performance ...

Also published in:
January 1999 **SIGPLAN Notices** Volume 34 Issue 1

5 An alternative approach to macro processing
Michael Hammer
December 1971 Proceedings of the international symposium on Extensible languages
Publisher: ACM

Full text available: 📄 Pdf (285.17 KB)   Additional Information: full citation, abstract, references, cited by, in
terms

Bibliometrics: Downloads (6 Weeks): 6,   Downloads (12 Months): 14,   Citation Count: 2

This paper presents the basic principles of, and the motivation for, an alternative me
of processing syntax macros. The inspiration for this work derives from some obvious
shortcomings and inefficiencies in conventional macro processing schemes. ...

Also published in:
December 1971 **SIGPLAN Notices** Volume 6 Issue 12

6 An intermediate language for source and target independent code optimization
Dennis J. Frailey
August 1979 **SI GPLAN '79:** Proceedings of the 1979 SIGPLAN symposium on Compiler
construction
Publisher: ACM
Full text available: 📄 Pdf (864.12 KB) Additional Information: full citation, abstract, references, index terms

Bibliometrics: Downloads (6 Weeks): 3,   Downloads (12 Months): 22,   Citation Count: 0

This paper describes an intermediate language to be generated by a syntax analyzer
processed by a code generator. An (essentially) optional code optimization phase ma
used before code generation. The language is designed to exclude source and ...

Also published in:
August 1979 **SIGPLAN Notices** Volume 14 Issue 8

7 Fully reflexive intensional type analysis
Valery Trifonov, Bratin Saha, Zhong Shao
September 2000 **I CFP '00:** Proceedings of the fifth ACM SIGPLAN international conferenc
Functional programming
Publisher: ACM ◇ Request Permissions

Full text available: Additional Information: full citation, abstract, references, cited by, in

Pdf (722.26 KB)   terms

Compilers for polymorphic languages can use runtime type inspection to support advanced implementation techniques such as tagless garbage collection, polymorphic marshalling, and flattened data structures. Intensional type analysis is a type-theoretic ...

Keywords: certified code, runtime type dispatch, typed intermediate language

Also published in:
September 2000 **SIGPLAN Notices** Volume 35 Issue 9

8 Approaches to design of high level languages for microprogramming
Patrick W. Mallett, T. G. Lewis
September 1974 **MICRO 7:** Conference record of the 7th annual workshop on Microprogramming
Publisher: ACM
Full text available: Pdf (605.30 KB) Additional Information: full citation, abstract, references, cited by, in terms

The development of a programming language for microprocessors (we will use the te microprocessor to refer to the hardware host to be microprogrammed) differs from th development of languages for conventional processors [5, 6]. The difference is traced

9 Towards a compiler front-end for Ada
Gerhard Goos, Georg Winterstein
December 1980 **SIGPLAN '80:** Proceedings of the ACM-SIGPLAN symposium on Ada programming language
Publisher: ACM Request Permissions
Full text available: Pdf (748.35 KB) Additional Information: full citation, abstract, references

This paper discusses the current development of a compiler front-end for Ada at the University of Karlsruhe. The front-end is independent of the target-machine and will compile Ada into an intermediate language AIDA, essentially an attributed structure .

Keywords: Ada, formal definition of Ada, intermediate language, semantic analysis, separate compilation, system programming language

Also published in:
November 1980 **SIGPLAN Notices** Volume 15 Issue 11

10 Deciding type equivalence in a language with singleton kinds
Christopher A. Stone, Robert Harper
January 2000 **POPL '00:** Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages
Publisher: ACM Request Permissions
Full text available: Pdf (1.40 MB) Additional Information: full citation, abstract, references, cited by, in terms

Work on the TILT compiler for Standard ML led us to study a language with singleton

kinds: S(A) is the kind of all types provably equivalent to the type A. Singletons are interesting ...

11 XIL and YIL: the intermediate languages of TOBEY

Kevin O'Brien, Kathryn M. O'Brien, Martin Hopkins, Arvin Shepherd, Ron Unrau
March 1995 Papers from the 1995 ACM SIGPLAN workshop on Intermediate representatic
Publisher: ACM

Full text available: Pdf (248.31 KB)   Additional Information: full citation, abstract, references, cited by, in: terms

Bibliometrics: Downloads (6 Weeks): 3,   Downloads (12 Months): 15,   Citation Count: 1

Typically, the choice of intermediate representation by a particular compiler implementation seeks to address a specific goal. The intermediate language of the TOBEY compilers, XIL, was initially chosen to facilitate the production of highly optim

Also published in:
March 1995 **SIGPLAN Notices** Volume 30 Issue 3

12 APL as a prototyping language: case study of a compiler development project

Matsuki Yoshino
December 1986 **APL '86:** Proceedings of the international conference on APL
Publisher: ACM

Full text available: Pdf (710.22 KB)   Additional Information: full citation, abstract, references, cited by, in: terms

Bibliometrics: Downloads (6 Weeks): 0,   Downloads (12 Months): 5,   Citation Count: 3

We are applying prototyping method using APL to a commercial compiler's developm Project. This paper will discuss the following matters based on our one year experien the project: Merits of APL as a prototyping language. An ...

Also published in:
May 1986 **SIGAPL APL Quote Quad** Volume 16 Issue 4

13 Flexible type analysis

Karl Crary, Stephanie Weirich
September 1999 **ICFP '99:** Proceedings of the fourth ACM SIGPLAN international confere on Functional programming
Publisher: ACM   Request Permissions

Full text available: Pdf (1.48 MB)   Additional Information: full citation, abstract, references, cited by, in: terms

Bibliometrics: Downloads (6 Weeks): 3,   Downloads (12 Months): 21,   Citation Count: 34

Run-time type dispatch enables a variety of advanced optimization techniques for polymorphic languages, including tag-free garbage collection, unboxed function arguments, and flattened data structures. However, modern type-preserving compile transform ...

Also published in:
September 1999 **SIGPLAN Notices** Volume 34 Issue 9

14 Current problems in automatic programming

Ascher Opler
May 1961 **IRE-AIEE-ACM '61 (Western):** Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference
Publisher: ACM

Full text available: Pdf (654.24 KB)   Additional Information: full citation, abstract

Bibliometrics: Downloads (6 Weeks): 0,   Downloads (12 Months): 12,   Citation Count: 0

The proliferation of the number of applications to be programmed and the number of types of computers available has created a significant and challenging problem. This survey will consider some of the more promising suggestions for enabling automatic

15 Functioning without closure: type-safe customized function representations for standard ML

Allyn Dimock, Ian Westmacott, Robert Muller, Franklyn Turbak, J. B. Wells

October 2001 **ICFP '01:** Proceedings of the sixth ACM SIGPLAN international conference Functional programming

Publisher: ACM ◆ Request Permissions

Full text available: Pdf (257.35 KB)   Additional Information: full citation, abstract, references, cited by, index terms

Bibliometrics: Downloads (6 Weeks): 2, Downloads (12 Months): 13, Citation Count: 3

The CIL compiler for core Standard ML compiles whole ML programs using a novel type intermediate language that supports the generation of type-safe customized data representations. In this paper, we present empirical data comparing the relative efficacy ...

Also published in:
October 2001 **SIGPLAN Notices** Volume 36 Issue 10

16 An overview of the PL.8 compiler

Marc Auslander, Martin Hopkins

June 1982 **SIGPLAN '82:** Proceedings of the 1982 SIGPLAN symposium on Compiler construction

Publisher: ACM

Full text available: Pdf (798.64 KB)   Additional Information: full citation, abstract, references, cited by, index terms

Bibliometrics: Downloads (6 Weeks): 6, Downloads (12 Months): 41, Citation Count: 50

The PL.8 compiler accepts multiple source languages and produces high quality object code for several different machines. The strategy used is to first do a simple translation of the source program to a low level intermediate language. Global optimization ...

Also published in:
June 1982 **SIGPLAN Notices** Volume 17 Issue 6

17 A flexible semantic analyzer for Ada

Mark S. Sherman, Martha S. Borkan

January 1980 Proceedings of the ACM-SIGPLAN symposium on The ADA programming language

Publisher: ACM

Additional Information: full citation, abstract, references, cited by, index terms

Bibliometrics: Downloads (6 Weeks): n/a, Downloads (12 Months): n/a, Citation Count: 5

A technique for writing semantic analysis phases of compilers is described. The technique uses Simula classes and virtual procedures to create a flexible and modular program. This technique is used to implement a semantic analysis phase of a compiler ...

18 Construction of a transportable, multi-pass compiler for extended Pascal

G. J. Hansen, G. A. Shoults, J. D. Cointment

August 1979 **SIGPLAN '79:** Proceedings of the 1979 SIGPLAN symposium on Compiler construction

Publisher: ACM

Full text available: Pdf (748.97 KB)   Additional Information: full citation, abstract, references, index terms

This paper describes the implementation of an extended Pascal compiler on the TI 99 minicomputer, the TI 980 minicomputer, and the IBM System 370. The compiler was designed to be as machine independent as possible; the parser and machine independent ...

Also published in:
August 1979 **SIGPLAN Notices** Volume 14 Issue 8

19 TCOLAda and the "Middle End" of the PQCC Ada compiler
Benjamin M. Brosgol
January 1980 Proceedings of the ACM-SIGPLAN symposium on The ADA programming language
Publisher: ACM
Additional Information: full citation, abstract, references, cited by, index terms

A compiler is traditionally partitioned into a (mostly) machine independent Front End which performs lexical, syntactic, and semantic analysis, and a machine dependent B End which performs optimization and code generation. In the Ada compiler being ...

20 Treat - an applicative code generator
Jerald S. Schwarz, Dean Rubine
January 1984 **POPL '84:** Proceedings of the 11th ACM SIGACT-SIGPLAN symposium on Principles of programming languages
Publisher: ACM Request Permissions
Full text available: Pdf (459.59 KB) Additional Information: full citation, abstract, references, index terms

Treat is a special purpose language for use in compiler writing. A Treat program translates a graph into c-trees (the intermediate language of the pcc compiler) and u the back end of the C compiler to generate code. Treat has been developed specifically ...